Optimisation

Master's degree in Modelling for Science and Engineering

Deterministic modelling.

Author:

Joseba Hernández Bravo

Teacher:

Lluís Alsedà

Barcelona, January 9, 2022

Contents

Chapter 1

Introduction

In this first section we give a brief introduction of the so-called *descent methods*. We will also introduce the two methods we will use to minimise the *Rosembroc's function*.

1.0.1 Descent methods

Descent methods are a set of iterative numerical methods used to find the minimum of a given function. They can be formulated as follows:

Given an initial point $\mathbf{x}^{(0)} \in \mathbb{R}$ for each $k \geqslant 0$ is iterated until convergence is obtained:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)} \tag{1.1}$$

As we can observe, these methods are based on two fundamental ideas. The first one is to find a value $\mathbf{d}$ which indicates the direction in which the function decreases the fastest. The second one is based on finding the step-size α between one point and the next.

In this work we will consider two different descent-methods in order find the minimum of the function 1.2:

- Steepest Descent Method.
- Conjugate Gradient Method.

1.0.2 Rosembroc's function

We define the Rosembroc's function as follows:

$$f(x, y) := a(y - x^2)^2 + (b - x)^2 \tag{1.2}$$

Where $a = 100$ and $b = 1$.

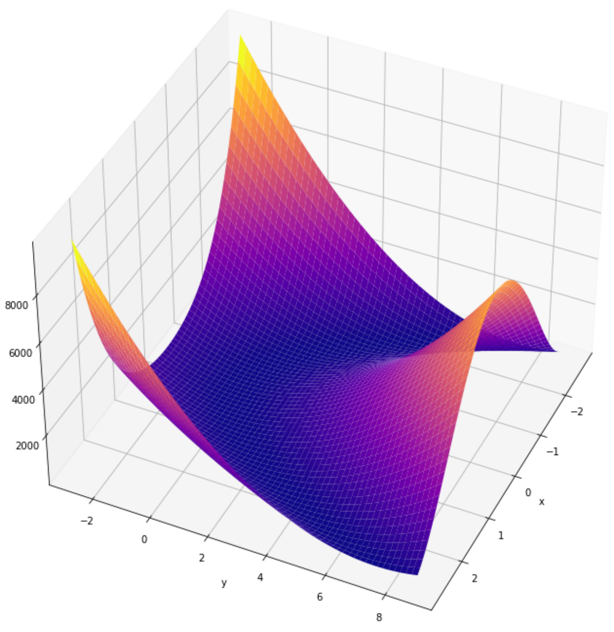


Figure 1.1: 3-D Ronsenbrock's function representation.

The function has the following properties:

- We say that f is continuously differentiable since it's partial derivative functions 1.3 exist and are continuous.

$$\begin{aligned} \frac{\partial f}{\partial x}(x, y) &= 2(200x^3 - 200xy + x - 1), \\ \frac{\partial f}{\partial y}(x, y) &= 200(y - x^2). \end{aligned} \tag{1.3}$$

  In addition, note that by equalising the two equations to 0 we find only one critical point. In this way we can deduce that we will have only one minimum/maximum. That is, the unique minimum/maximum of the function will at the same time be the global minimum.

- Since the Rosenbrock's function is a continuously differentiable function, we can obtain it's **global minimum** analytically. Furthermore, as the function is positively defined in all it's domain, we can calculate the minimum by setting it to zero. Thus,

$$f(x,y) := 100(y-x^2)^2 + (1-x)^2 = 0 \iff x = 1 \text{ and } y = 1.$$

- Since the the function $f : \mathbb{R}^2 \to \mathbb{R}$ is twice continuously differentiable with **dom** f open, then we say that f is convex in **dom** f if and only if the Hessian matrix of f is positive semidefinite for any x in **dom** f. i.e.,

$$f \text{ is convex} \iff H(f) \text{ is positive semidefinite } \forall x \in \mathbf{dom} f$$

Therefore, by calculating the second partial derivatives of the function 1.2, we have that:

$$H(f) = \begin{bmatrix} 400(3x^2-y)+2 & -400x \\ -400x & 200 \end{bmatrix}. \tag{1.4}$$

By definition $H(f)$ is positive semidefinite if

$$\mathbf{x}^t H(f)\mathbf{x} = 200(x^4 - 6x^2y + y^2) + 2x^2 \geqslant 0.$$

As we can see, the above condition is not satisfied for all values of x and y. Therefore, we can say that the function is **not convex** over the whole domain of f.

In the next image we can see the values of x and y in which f is not convex:

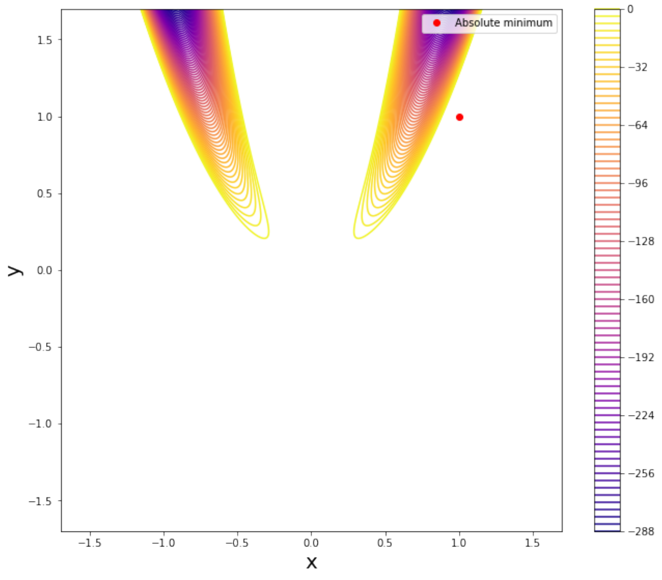


Figure 1.2: values of x and y for which the function f is non-convex. i.e. values for which $\mathbf{x}^t H(f)\mathbf{x} < 0$.

1.0.3 Seed

One of the first questions we can ask ourselves when we try to find the minimum of a function using algorithms such as the ones we will use in this work is the following: are we using a good starting point?

By looking at the results of the following sections we will see that the algorithm works fine. Therefore, as a first conclusion, we can say that the initial point $(-1.5, -1)$ is valid and that it is acceptable. How can we make the algorithm converge faster?

Let us imagine that we introduce small perturbations in the parameters a and b of the equation 1.2. By doing so, we will observe that the form of the function does not change drastically, which indicates that the minimum in these cases will be close to the point $(x, y) = (1, 1)$. Therefore, a good strategy to follow would be to observe the zones close to this point with the highest convexity.

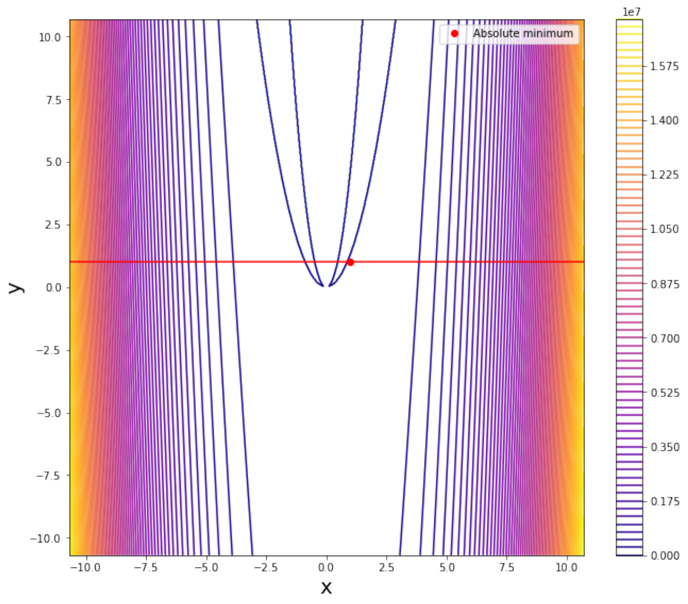


Figure 1.3: Values of x and y for which the function f is convex.

As we can see in the image, the areas of greater convexity are those corresponding to the left and right sides. Therefore, in the case that we are looking for a minimum around the point (1,1) we can start by drawing a horizontal line that crosses this point (red line in the image).

Given that the part of the red line coming from the left-hand side passes through areas of little or no convexity, we can deduce that the areas from which the algorithm will converge first are those corresponding to the right-hand side.

We can therefore deduce that the algorithm will converge earlier in cases where the initial value takes approximately the following values: $y \in [0, 2.5]$ and $x > 2.5$.

Chapter 2

Programme structure and results

In this section we will explain the steps followed for the solution of the problem presented in the previous section. For this, as we have explained previously, we will rely on two different methods: Steepest Descent Method and Conjugate Gradient Method.

The whole algorithm has been implemented in a single file called `deterministic.c` and has been stored at the following address: GitHub link

The program created, consists of a total of 6 functions in charge of performing different calculations such as the value of the step-size α or the direction $\mathbf{d}$ from 1.2 and a main function that controls whether the programme has converged or not. The convergence criterion has been established for a value of the derivative equal to or less than $1 \cdot 10^{-9}$.

Since both methods have been applied for the same function, and furthermore, both use the partial derivatives of the same function to calculate the direction d, we have created two common functions called so and so.

On the other hand, another function that has been used in a common way is `SteepestDescent()` which does the step-size calculation. This, as we have explained in the previous section, is one of the most important tasks to achieve an efficient convergence of our problem. Here we have implemented the *backward method*, which is a line search method (an iterative), with an starting value of $\alpha_k = 1$ and that terminate as soon as some accuracy stopping criterion is satisfied. So, following the same notation as in Quarteroni et al., 2010, we have established a different values of σ[1] $\in (0, 1/2)$ and ρ[2] $\in (0, 1)$ for each of the method used. These are the different rho and

[1] A constant used in the stoping criterion

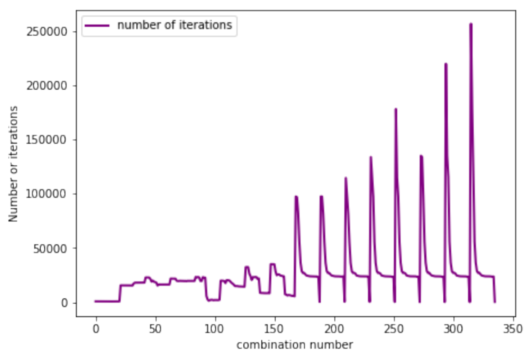
[2] The backward-step. For each iteration until the established convergence criterion is met, we multiply the value of α by this value.

sigma values used:

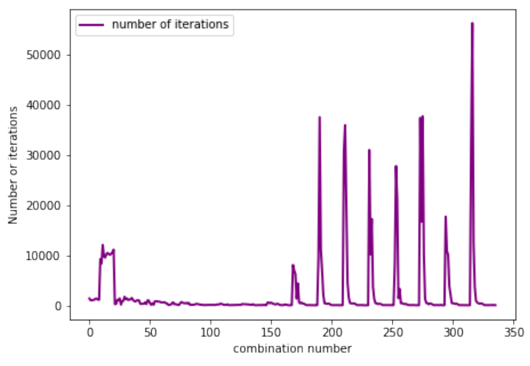
```
const double sigmavec[] =
    {0.0001,0.001,0.0023,0.01,0.03,0.06,0.09,0.1,0.13,0.16,0.26,0.36, 0.49,
0.491,0.492,0.493,0.494,0.495,0.497,0.496, 0.5};

const double rhovec[] =
    {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.89,0.99,0.991,0.992,0.993,0.994,0.995, 0.996,
    0.997};
```

In this way, by setting a constant ρ value for each iteration, we have evaluated the number of iterations for each of the σ values.



(a)



(b)

Figure 2.1: Number or iterations in gradient descent (a) and number of iterations in conjugate gradient descent (b) vs a pair of the correspondent $[\rho, \sigma]$.

The peaks of high iterations correspond to pairs of values where σ takes a very small value while ρ takes values around 1.

2.0.1 Steepest Descent Method

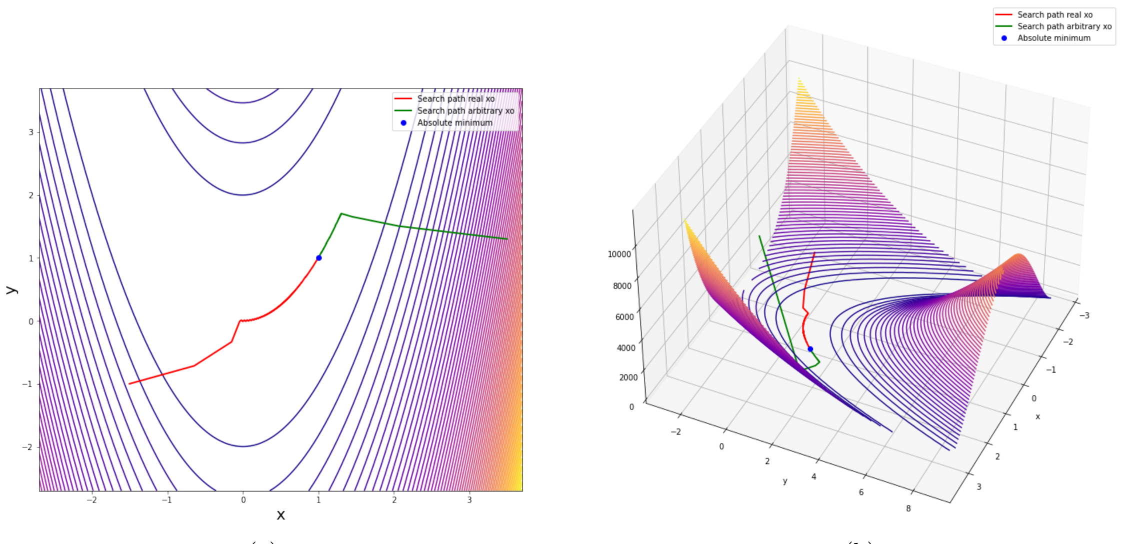
For the steepest descent method, the direction in which we move takes the value of $\mathbf{d} = -\nabla f(x_k)$. Therefore equation 1.1 is rewritten as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha(-\nabla f(x_k)) \tag{2.1}$$

As we have explained, after searching through different values of σ and ρ, we have found the optimal values for 0.499 and 0.99 respectively. So that:

Starting point	ρ	σ	**Nº of iterations**
(-1.5,-1)	0.994	0.500	273
(3.5,1.3)	0.994	0.500	198

Moreover, the trajectories followed in both cases have been as follows



(a) (b)

Figure 2.2: Trajectories for the initial points $(x, y) = (-1.5, -1)$ and $(x, y) = (3.5, 1.3)$ in 2-D (a) and in 3-D.

2.0.2 Conjugate Gradient Method

In this second case, we define the direction $\mathbf{d}$ as follows:

$$d_k = -\nabla f(x_k) + \beta_k \mathbf{d}_{k-1} \tag{2.2}$$

Where $\beta = \left(\frac{||\nabla f(x_k)||}{||\nabla f(x_{k-1})||}\right)^2$ when using the *Fletcher–Reeves* method Luenberger, Ye, et al., 1984.

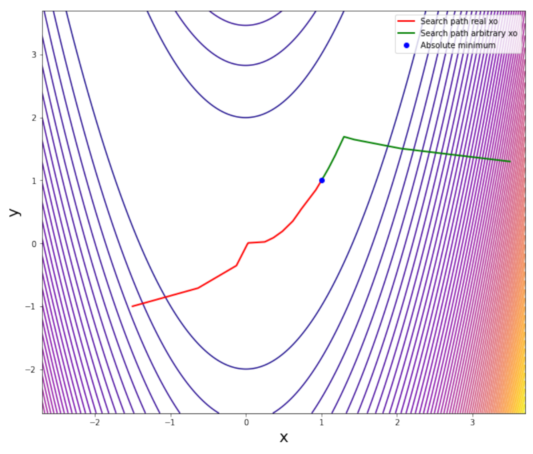
Therefore equation 1.1 is rewritten as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha(-\nabla f(x_k) + \beta_k \mathbf{d}_{k-1}) \tag{2.3}$$

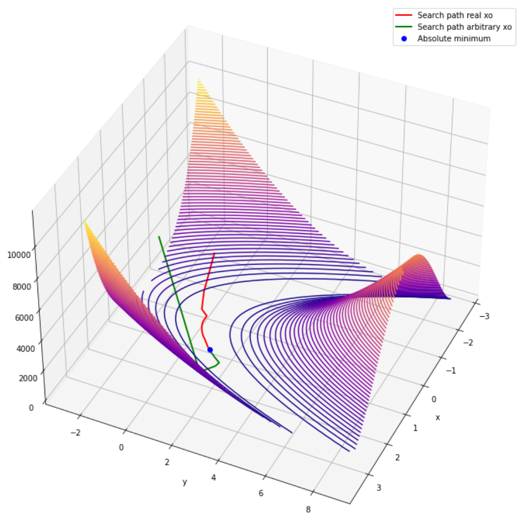
For this second case we have applied the same procedure as for the **Gradient descent method**. Thus, we have found the best result in the case in which $\sigma = 0.991000$ and $\rho = 0.495000$. So that:

Starting point	ρ	σ	**Nº of iterations**
(-1.5,-1)	0.495	0.500	31
(3.5,1.3)	0.495	0.500	26

As in the previous case, we show the trajectories followed for the two cases with different starting points:



(a)



(b)

Figure 2.3: Trajectories for the initial points $(x, y) = (-1.5, -1)$ and $(x, y) = (3.5, 1.3)$ in 2-D (a) and in 3-D (b).

Chapter 3

Conclusions

In this work we have used the *steepest descent* and *conjugate gradient descent* methods in order to minimise the function presented in 1.0.2. Both methods have successfully converged to the minimum of the function (previously calculated analytically).

In the case of the *steepest descent* we have iterated the algorithm using different constant ρ and σ values. In this way, we have obtained the optimal value in the case where $\sigma = 0.500$ and $\rho = 0.994$. Using these values and with a convergence criterion set to $\nabla f(x, y) = 1 \cdot 10^{-7}$, we found the minimum value of the function in 273 iterations. We have also observed that the critical points for this algorithm are the points where the slope is very low. Particularly in the middle zone of the images presented in the figure 2.2 , it can be observed that the algorithm makes shorter and shorter steps until the solution is finally found.

In the case of the *conjugate gradient descent*, we have followed exactly the same procedure as in the previous one. We have selected the optimal parameters sigma and rho and we have calculated the number of iterations necessary to reach the minimum. It should be noted that the convergence criterion in both cases has been the same. In this case, however, we have obtained a total of 36 iterations, approximately 8.5 times less than with the first method. This shows that for areas where the slope is not very steep, the second method works much better.

On the other hand, we have performed the same procedure using a different starting point. Due to the shape of the function, we have deduced that the most suitable a priori values for an accurate and fast operation of the algorithms should be those corresponding to the zone where $x > 2.5$ and $y \in [0, 2.5]$. Thus, we have established the new seed in $(x, y) = (3.5, 1.3)$. After setting the values of ρ and σ exactly as in the previous cases, it can be said that we have obtained a notable reduction in the number of iterations. In the first case, the iterations have been reduced from 273 to 198 while in the second case the iterations have dropped from 32 to 26.

Bibliography

Luenberger, D. G., Ye, Y. et al. (1984). *Linear and nonlinear programming* (Vol. 2). Springer.

Quarteroni, A., Sacco, R., & Saleri, F. (2010). *Numerical mathematics* (Vol. 37). Springer Science & Business Media.